MICROCONTROLADOR 8051

NOTA IMPORTANTE: Esta documentação está em processo de revisão. Estamos trabalhando duro para assegurar que todas as informações contidas neste documento estejam corretas. Não nos responsabilizamos por qualquer dano ou perda provocado pelo uso deste material.

---

1 - Introdução aos Microcontroladores

Inicialmente, é importante frisarmos três conceitos:

a) ***Microprocessador***: chip responsável pelo processamento em um microcomputador. É um elemento complexo, contendo, entre outras coisas, uma unidade lógica e aritmética (ULA) e diversos registros (registradores) especiais.

b) ***Microcomputador-de-um-só-chip***: como o nome indica, reúne no mesmo chip os diversos elementos de um microcomputador: microprocessador, RAM, ROM, temporizadores, contadores, canal de comunicação serial e portas de I/O.

c) ***Microcontrolador***: microcomputador-de-um-só-chip que pode ainda conter elementos para uso industrial, tais como conversores A/D e D/A, PLL, PWM, etc.

1.1 – Motivação para o Estudo de um Microcontrolador

O microcontrolador, hoje em dia, é um elemento indispensável para o engenheiro elétrico, eletrônico ou ainda para o técnico de nível médio da área, em função de sua versatilidade e da enorme aplicação. Entre algumas das aplicações de um microcontrolador podemos citar automação industrial, telefones celulares, auto-rádios, fornos de microondas e videocassetes. Além disso, a tendência da eletrônica digital é de se resumir a microcontroladores e a chips que concentram grandes circuitos lógicos, como os PLDs (Programmable Logic Devices). Para sistemas dedicados, o microcontrolador apresenta-se como a solução mais acessível, em função do baixo custo e facilidade de uso.

1.2 – O 8051

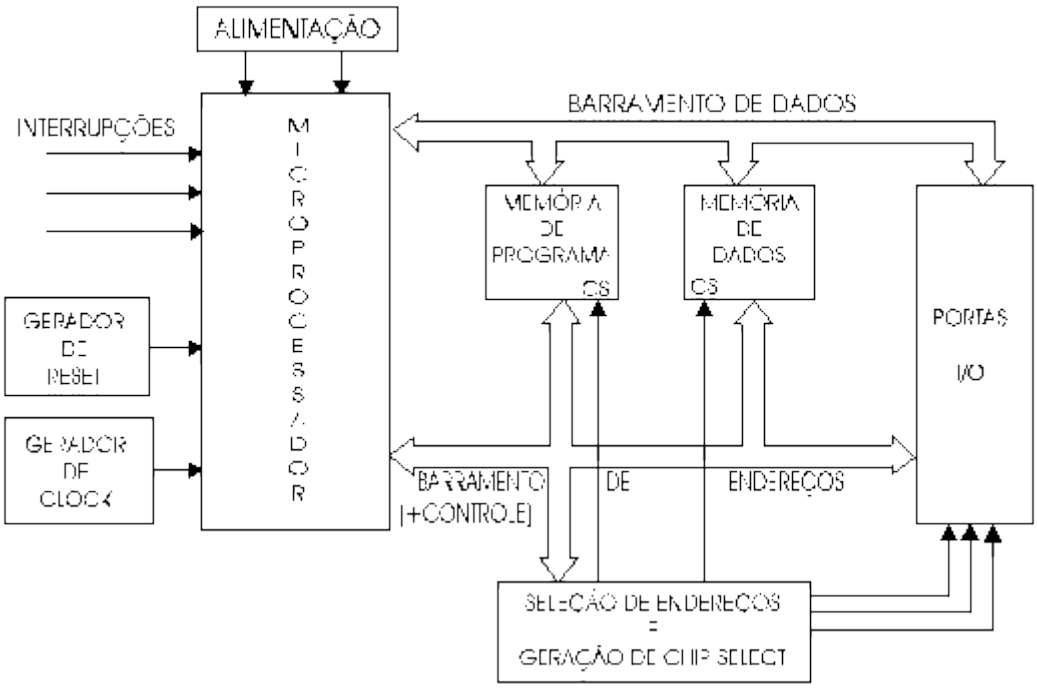
O 8051, da Intel, é, sem dúvida, o microcontrolador mais popular atualmente. O dispositivo em si é um microcontrolador de 8 bits relativamente simples, mas com ampla aplicação. Porém, o mais importante é que não existe somente o CI 8051, mais sim uma *família* de microcontroladores baseada no mesmo. Entende-se *família* como sendo um conjunto de dispositivos que compartilha os mesmos elementos básicos, tendo também um mesmo conjunto básico de instruções.

2 - Sistemas Microprocessados

Sistemas microprocessados são aqueles que têm por elemento central um microprocessador. O microprocessador funciona como um sistema seqüencial síncrono, onde a cada pulso, ou grupos de pulsos de clock, uma instrução é executada. Entre os microprocessadores mais conhecidos podemos citar o 8080 e 8085, Z-80, 8088, 8086, 80286, 68000, 80386 e superiores.

Embora já existam microprocessadores que trabalhem a centenas de MHz, o 8051 utiliza tipicamente um clock de 12 MHz, com tempos de execução de cada instrução variando entre 1µs e 4µs.

2.1 – Diagrama em Blocos de um Sistema Genérico com Microprocessador



Além do microprocessador, um sistema básico como este tem os seguintes elementos:

Interrupções: são entradas a partir de um sinal externo que fazem com que o processamento seja interrompido e seja iniciada uma subrotina específica. (Obs.: o 8051 tem interrupções com estrutura *nesting*, onde uma interrupção pode interromper outra que está sendo atendida, desde que tenha maior prioridade).

Gerador de Reset: responsável por inicializar o sistema ao ligar ou quando acionado.

Gerador de Clock: gera os pulsos necessários ao sincronismo do sistema.

Memória de Programa: memória onde o microprocessador vai procurar as instruções a executar. Em sistemas dedicados costumam-se utilizar memórias ROM, embora em alguns casos memórias RAM também sejam utilizadas.

Memória de Dados: memória onde o microprocessador lê e escreve dados durante a operação normal. Geralmente é do tipo volátil, embora memórias não-voláteis possam ser utilizadas.

Seleção de Endereços: lógica para escolher qual memória ou periférico o microprocessador vai utilizar.

Portas de I/O: sua função é a comunicação com o mundo externo. Através delas dispositivos como teclados, impressoras, displays, entre outros, comunicam-se com o sistema.

3 - Hardware

Como já foi citado, o 8051 é um microcontrolador de ampla utilização. O mesmo tem dois modos básicos de funcionamento:

a) modo mínimo, onde somente recursos internos são utilizados pela CPU. Neste modo, estão disponíveis 4 KB de ROM para memória de programa e 128 bytes de RAM para memória de dados. O modo mínimo possui a vantagem (além da economia de componentes e espaço físico) de poder utilizar as 4 portas de 8 bits cada para controle (I/O);

b) modo expandido. Neste modo, a memória de programa (ROM), a memória de dados (RAM) ou ambas podem ser expandidas para 64 kB, através do uso de CIs externos. No entanto, apresenta a desvantagem de "perder" duas

das 4 portas para comunicação com as memórias externas.

A pinagem para o 8051, é mostrada abaixo na figura 3.1:

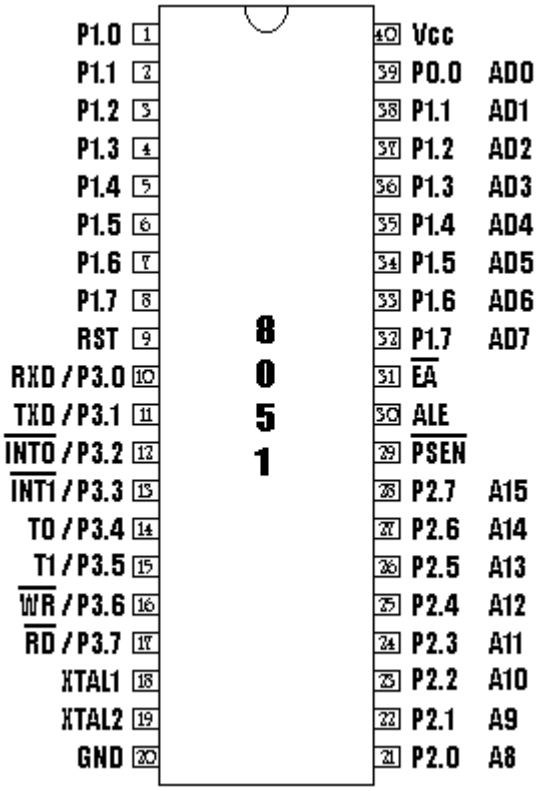


Figura 3.1

Um exemplo de utilização do modo expandido é mostrado na figura 3.2. No caso, estão sendo utilizadas externamente tanto memória de programa como memória de dados, mas é possível a utilização de apenas uma delas (somente a ROM).

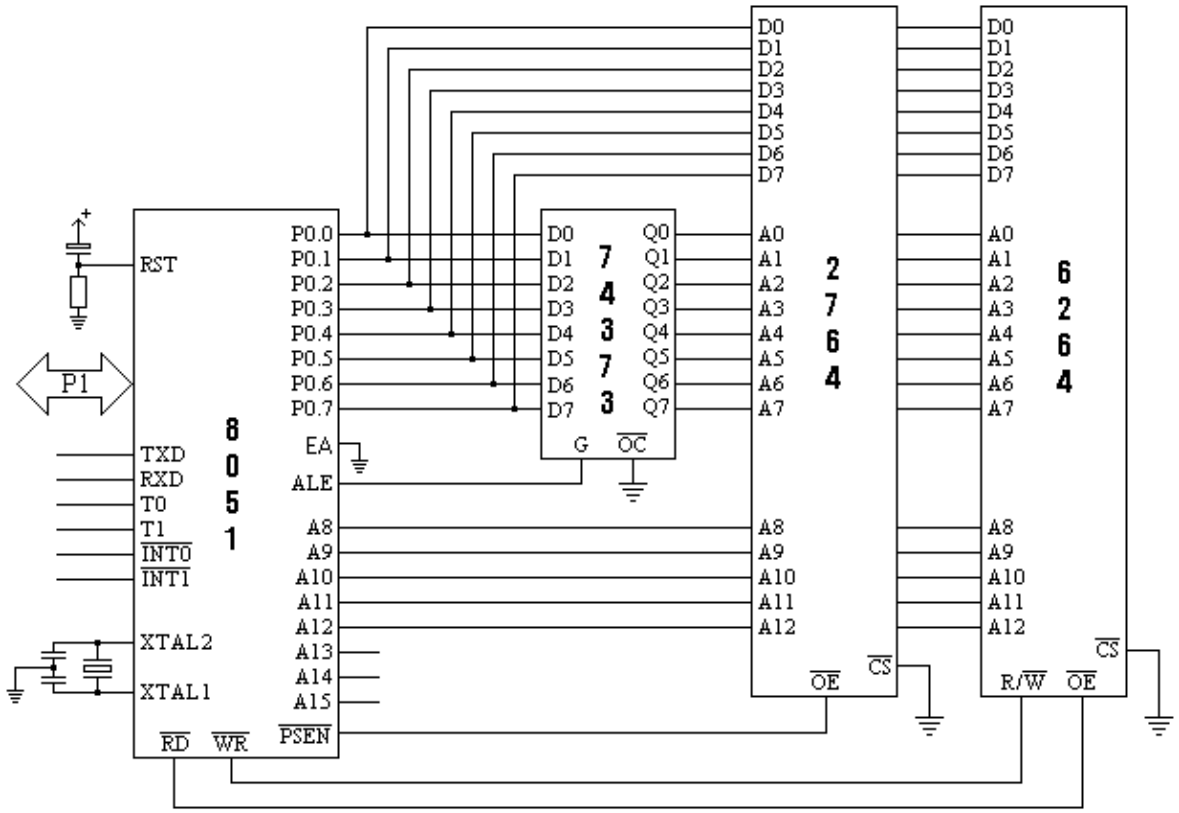


Figura 3.2 - Expandindo RAM e ROM

Como pode ser visto na figura 3.2, além das memórias faz-se necessário a utilização de um outro CI (no caso o 74373) para a multiplexação de dados e endereços. Ocorre da seguinte forma: no primeiro ciclo de máquina, o 8051 coloca nos pinos AD0 a AD7 o byte menos significativo do endereço externo e leva o pino ALE (*Address Latch Enable* - Habilitador da Trava de Endereços) a nível alto, de modo que o 74373 (oito Latches tipo D) coloque em suas saídas essa informação, e logo em seguida passa este pino para nível baixo, para que esse byte fique retido no 74373. Após isso, os pinos AD0 a AD7 estarão livres para o transporte dos dados.

O 8051 pode, no modo expandido, utilizar toda a memória de programa externa (com nível lógico 0 aplicado ao pino 31 - External Address NOT) ou ainda utilizar os primeiros 4 kB internos e o restante externo (com 1 em EA).

O **CI 8031** é a versão *sem ROM interna* do 8051. O mesmo é muito utilizado em fase de desenvolvimento ou quando se quer produzir em pequenas quantidades. Como o 8031 tem a mesma pinagem que o 8051, o mesmo possui o pino EA, que deverá **sempre** ser utilizado em nível lógico baixo.

Retornando a figura 3.2, nela estamos utilizando 8kB de RAM externa, além dos 256 bytes de RAM interna. Além disso, temos um **total** de 8kB de memória ROM que, no caso do 8051, pode estar sendo utilizado apenas no CI externo ou com os 4096 bytes menos significativos em memória interna e os 4096 bytes mais significativos na memória externa.

Ainda em relação à figura 3.2, para fazer uma leitura na ROM externa, o pino **PSEN** será levado a nível baixo, para fazer uma leitura na RAM externa, o pino **RD** será levado a nível baixo e, para fazer uma escrita na RAM externa, o pino **WR** será levado a nível baixo.

4 - A Organização da Memória

As figuras 4.1 e 4.2 ilustram, respectivamente, a organização da memória de programa e da memória de dados do 8051.

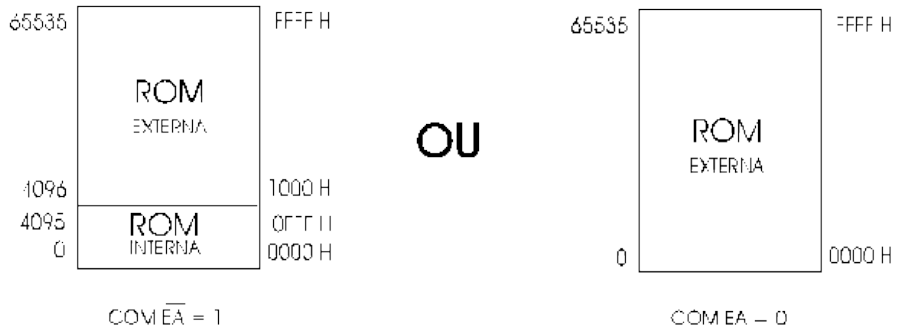


Figura 4.1 - Memória de Programa

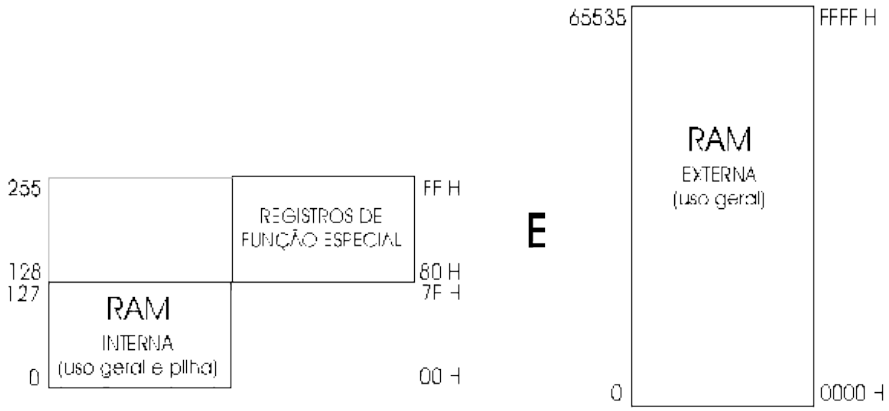


Figura 4.2 - Memória de Dados

É importante salientar que, diferentemente de outros sistemas baseados em microprocessador, onde cada endereço de memória identifica uma única posição física, no 8051 o mesmo endereço hexadecimal pode identificar 3 posições físicas diferentes (e até 4, no caso do 8052, que tem 256 bytes de RAM interna). Por exemplo, temos o endereço 23H na RAM interna, o endereço 23H na RAM externa e o endereço 23H na ROM externa. Mesmo com esses endereços "iguais", não há risco de confusão, pois as instruções e o modo de endereçamento são diferentes. Para deixar mais claro, vamos supor que queiramos carregar no acumulador o conteúdo de cada uma das posições 23H. Veja os exemplos abaixo:

a) da RAM interna

```
MOV A,23H ;end.direto
```

b) da RAM externa

```
MOVX R0,#23H ;imediato
;R0 como ponteiro

MOVX A,@R0 ;indireto
```

c) da ROM externa

```
CLR A

MOV DPTR,#0023H

MOVC A,@A+DPTR
```

A figura 4.3 mostra em detalhes a divisão da RAM interna do 8051. Note a presença dos 4 bancos de registradores auxiliares, cada um contendo 8 registradores (R0, R1, R2, R3, R4, R5, R6, R7). Estes registradores são utilizados para endereçamento indireto. Apesar de existirem 4 bancos de registradores, somente um está ativo por vez para uso como índice. A seleção do banco ativo é feita no registro de função especial PSW, que será visto em breve.

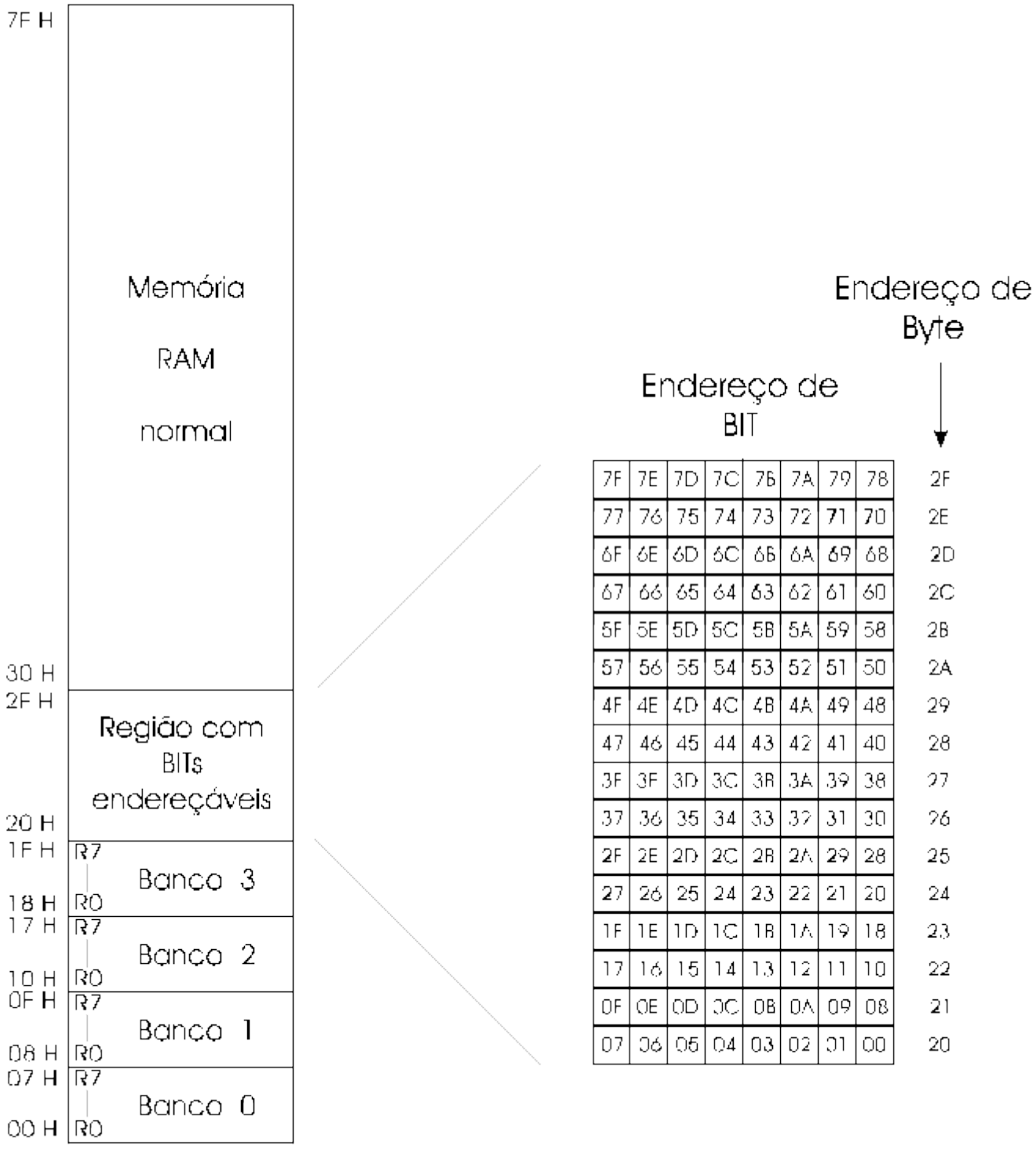


Figura 4.3 - RAM Interna

Como pode ser visto pela figura, os **bytes** 20 a 2F da RAM interna tem bits endereçáveis individualmente. Podemos, com os mesmos, executar várias instruções de bits. Por exemplo, a instrução **SETB 3CH** coloca em nível alto o bit 3C ou seja, o bit 4 da posição de memória 27 H da RAM interna.

5 - Os Registros de Função Especial

Os Registros de Função Especial (SFRs - *Special Function Registers*) são responsáveis pela maior parte do controle do 8051. Os mesmos são mostrados na figura 5.1, sendo que alguns deles possuem bits endereçáveis. Note que alguns dos bits endereçáveis possuem inclusive um nome mnemônico, para maior facilidade de desenvolvimento de software em compiladores.

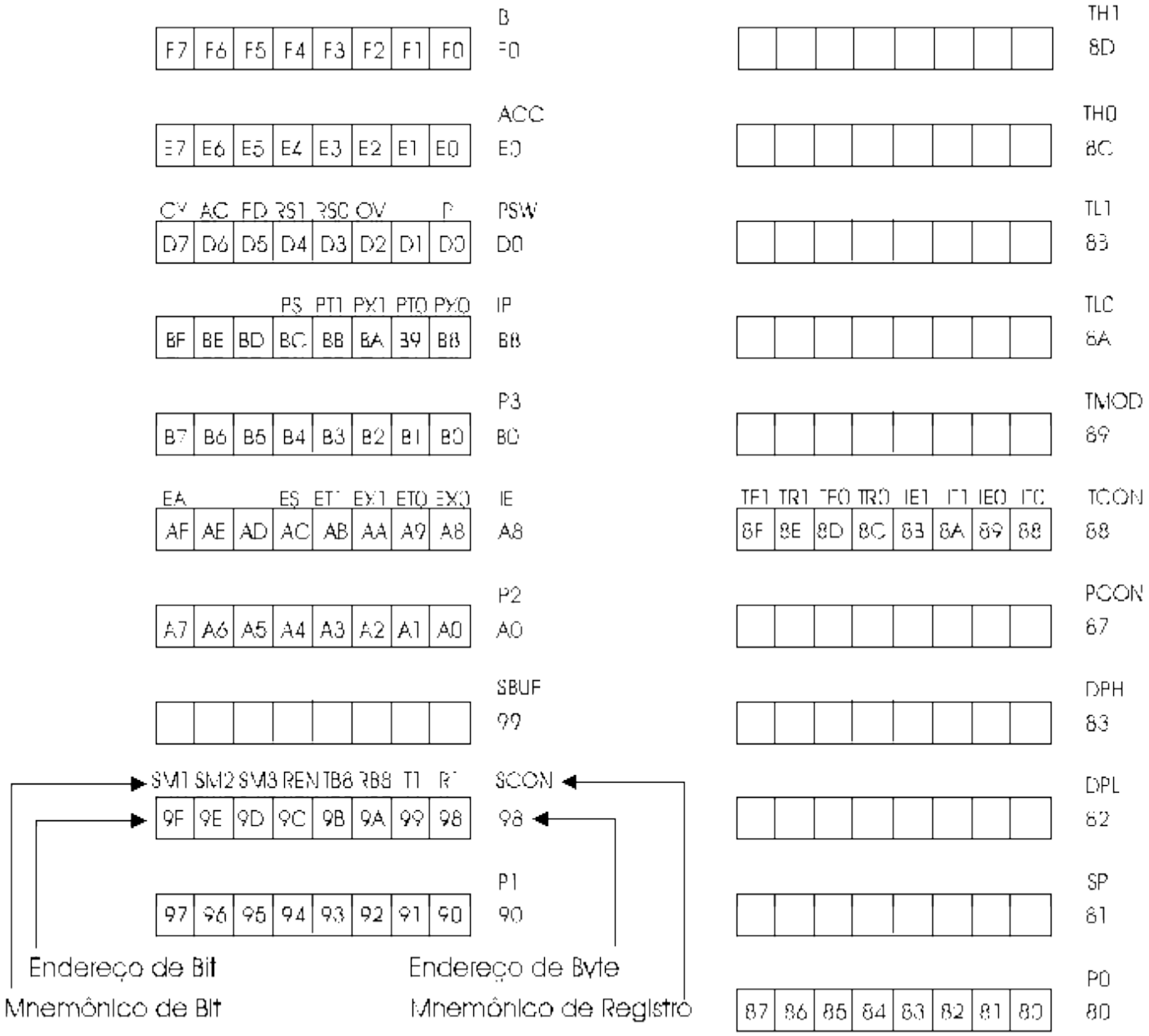


Figura 5.1

A função básica de cada registro especial é descrita abaixo. Ao lado do registro está seu endereço na RAM.

P0 (80H), **P1** (90H), **P2** (A0H) e **P3** (B0): São as posições na RAM que contém os dados das 4 portas de I/O do microcontrolador, caso as mesmas sejam utilizadas com esse fim. Se for feita uma escrita em um desses registros, o conteúdo nos pinos correspondentes à porta será automaticamente alterado. Já uma leitura dos mesmos verifica o estado dos pinos.

SP (81H): É o apontador de pilha (*Stack Pointer*), que aponta para o alto da pilha.

DPL (82H) e **DPH** (83H): Em conjunto formam o DPTR, utilizado para endereçamento indireto de 16 bits.

PCON (87H): O registro PCON (*Power Control*) permite adaptar o microcontrolador para redução de consumo com segurança.

TCON (88H) e **TMOD** (89H): Registros dos Temporizadores/Contadores. Permitem a programação dos mesmos.

TL0 (8AH), **TH0** (8CH), **TL1** (8BH) e **TH1** (8DH): São os registros dos dados dos dois Temporizadores/Contadores (T0 e T1).

SCON (98H) e **SBUF** (99H): Registros para uso da porta de comunicação serial.

IE (A8H) e **IP** (B8H): Registro para programação (habilitação/desabilitação, prioridade etc.) das interrupções.

PSW (D0H): O PSW (*Program Status Word* - palavra de status do programa) é o registro dos Flags do 8051

ACC (E0H): É o acumulador.

B (F0H): Registro auxiliar B.

A seguir, são mostrados com mais detalhes alguns dos registros de função especial.

5.1 – Palavra de Status de Programa (P S W)

A palavra de status de programa (PSW) contém vários bits de status que refletem o estado corrente da CPU.

O registro PSW é acessado pelo endereço D0h e também é bit endereçavel.

A figura abaixo apresenta a estrutura do registro PSW, que reside no espaço de endereçamento de registros de função especial.

D7h	D6h	D5h	D4h	D3h	D2h	D1h	D0h
CY	AC	F0	RS1	RS0	OV	-	P
PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PWS.1	PSW.0

PSW.7 - Flag de carry para bit 7 da ALU;

PSW.6 - Flag de carry auxiliar para bit 3 da ALU (para operações BCD);

PSW.5 - Flag 0 - Flag de status de propósito geral (definido pelo usuário);

PSW.4 - Bit 1 de seleção de banco

PSW.3 - Bit 0 de seleção de banco

PSW.2 - Flag de overflow para operações aritméticas;

PSW.1 -

PSW.0 - Flag de paridade do acumulador (1=ímpar, 0=par).

Este registro contém o bit de carry, carry auxiliar, para operações BCD, dois bits de seleção de banco de registros, flag de overflow, bit de paridade e dois flags de status definidos pelo usuário.

O bit de carry é usado em operações aritméticas e, também, serve como "acumulador" para operações booleanas.

Os bits RS0 e RS1 são usados para selecionar um dos quatro bancos de registros mostrado na figura 4.3 (pg.6). As instruções referem-se a estas localizações de RAM como registros de R0 a R7. A seleção de qual dos quatro bancos estará sendo utilizado é feita com base aos conteúdos de RS0 e RS1, durante o tempo de execução do programa.

RS1	RS0	BANCO	Endrereço na RAM interna
0	0	0	00 h - 07 h
0	1	1	08 h - 0F h
1	0	2	10 h - 17 h
1	1	3	18 h - 1F h

O bit de paridade reflete o número de 1's do acumulador. P = 1 se o acumulador tiver um número ímpar de 1's e P = 0, se tiver um número par. Assim o número de 1's do acumulador mais P será sempre par.

O flag de overflow é afetado por operações aritméticas.

Além do flags usuais, como CY, AC , OV, P, tem dois flags de propósito geral que não estão associados com qualquer estado ou função da CPU. O bit PSW.5 (F0) e PSW.1 (sem nome). O programador pode usar F0 (PSW.5) como um bit flag para aplicação definida pelo usuário. Pode ser setado ou ressetado pelo programa como uma função de alguma condição especial ou ser lido de um pino de alguma porta. Documentos da INTEL indicam que PSW.1 esta reservado para uma utilização futura e é recomendado que não seja utilizado em programas

Note que o PSW não tem flag ZERO, mas isto não é problema porque o 8051 tem instrução específicas para testar se o acumulador é zero.

Exemplos: JZ rel. - desvia se o acumulador for zero

JNZ rel - desvia se o acumlador não for zero

* rel. = indica um byte sinalizado com complemento de dois, usado para deslocamento relativo na faixa de –127 a +128.

5.2 – Registro Apontador de Pilha (SP)

O registro SP é um registro de tamanho 8 bits. Este será incrementado antes que um dado seja armazenado na pilha, durante a execução de uma instrução PUSH, ou CALL.

Assim, a região de pilha (stack) irá residir no espaço de RAM interna do 8051. O SP é inicializado com o valor 07 H após ocorrer o RESET. Isto faz com que a pilha inicie na localização 08 H. Caso sejam usados os bancos de resgistradores 1, 2 e/ou 3, e seja utilizada a pilha, o SP deverá ser alterado para iniciar a pilha em outra região da RAM que não cause problemas. A intrução MOV IE,*#novo endereço* deverá aparecer no programa, preferencialmente no início.

Este é acessado pelo endereço 81 H e portanto não é bit endereçável.

5.3 – Registro Apontador De Dados (DPTR)

O registro apontador de dados é formado por um byte alto (DPH) acessado pelo endereço 83 H e por um byte baixo (DPL), acessado pelo endereço 82 H. Este registro pode ser manipulado como um registro de 16 bits (DPTR), ou como dois registros independentes de 8 bits.

A sua função é de manter um endereço de 16 bits, usado para acessar memórias externas.

6 - MODOS DE ENDEREÇAMENTO

Como o conjunto de instruções da família 8051 foi otimizado para aplicação de controle em 8 bits. Contém uma variedade de rápidos modos de endereçamento para acessar a RAM interna, facilitando operações de byte em pequenas estruturas de dados.

6.1. – Modo de endereçamento DIRETO

Neste modo o operando é especificado na instrução por um campo de endereço de 8 bits. Somente a RAM de dados interna e os registros de função especial (primeiras 256 posições de memória) é que poderão ser endereçados diretamente.

Exemplo 6.1.a : MOV A, 25h

A ← (25h)

Exemplo 6.1.b : ADD A, 7Fh

A ← A + (7F)

Exemplo 6.1.c : MOV 90h, A Obs.: 90h = porta 1

(90h) ← A

6.2 – Modo de endereçamento de registro ou modo REGISTRADOR

Os bancos de registros, contendo de R0 a R7, serão acessados por certas instruções onde a especificação do registro será feita por três bits do próprio opcode. As instruções de acesso aos registros são eficientes, visto que nenhum endereço será necessário. Quando o instrução for executada, um dos oito registros do banco selecionado será acessado. Um dos quatro bancos de registro será selecionado pelos bits de seleção de bancos do registro PSW (bits RS1 e RS0).

Exemplo 6.2.a : MOV R5, A

R5 ← A

Exemplo 6.2.b : ADD A, R0

A ← A + R0

6.3 – Modo de endereçamento INDIRETO

Neste modo a instrução especifica que o registro contém o endereço do operando. Tanto a memória interna quanto a externa poderão ser endereçadas indiretamente.

O registro de endereçamento usado para endereços de 8 bits deverá ser o registro R0 ou R1 do banco selecionado e para endereços de 16 bits será somente o registro *apontador de dados* (DPTR).

Exemplo 6.3.a : MOV @ R1, 15h

(R1) ← (15h)

Exemplo 6.3.b : ADD A, @ R0

A ← A + (R0)

Exemplo 6.3.c : MOVX @ DPTR, A

(DPTR) ← A

Obs.: @ é utilizado para indicar endereçamento indireto.

@ = endereçado pelo conteúdo de ...

6.4 – Modo de endereçamento de registros específicos ou ESPECÍFICO A REGISTRO

Algumas instruções referem-se a certos registros. Por exemplo algumas instruções operam o acumulador, o registro DPTR, etc., assim nenhum byte de endereço será necessário, o opcode já define qual o registro que será afetado.

Exemplo 6.4.a : DA A

Faz o ajuste decimal do acumulador

Exemplo 6.4.b : CLR A

A ← 00h (zera o acumulador)

Exemplo 6.4.c : INC DPTR

DPTR ← DPTR + 1

6.5 – Modo de endereçamento IMEDIATO ou CONSTANTE IMEDIATA

Neste modo de endereçamento o opcode é seguido de um valor de uma constante que será operada. Na linguagem assembly, este modo é indicado através do simbolo #.

Exemplo 6.5.a : MOV B, #252

B ← FCh

Exemplo 6.5.b : MOV A, #100

A ← 64h

Exemplo 6.5.c : MOV DPTR, #05FEh

DPTR ← 05FEh

Obs.: # indica valor constante;

Quando após a constante aparecer um "*H*", o valor da constante é *hexadecimal*, quando tiver um "*B*" é *binário,* e quando a letra for omitida ou aparecer um "*D*" o valor será *decimal.*

6.6 – Modo de endereçamento INDEXADO

Somente a memória de programa (ROM) poderá ser acessada com endereçamento indexado e somente poderá ser lida.

O endereço efetivo é a soma do acumulador e um registro de 16 bits (DPTR ou PC).

Este modo é usado para leituras de tabelas colocadas na memória de programa (ROM). Por exemplo tabelas de conversão, ou de mensagens.

Um registro base de 16 bits, tal como o registro DPTR, ou contador de programa (PC), aponta para a base da tabela e o acumulador recebe o deslocamento dentro da tabela. Assim o endereço de entrada da tabela será formado com a soma do conteúdo do acumulador e o registro base.

Exemplo 6.6.a : MOVC A, @ A + DPTR

A ← (A+DPTR) da ROM

Exemplo 6.6.b : MOVC A, @ A + PC

A ← (A + PC) da ROM

Outro tipo de endereçamento indexado é usado nas instruções de "case jump". Neste caso o endereço destino do salto (jump) é calculado com a soma do conteúdo do acumulador e do conteúdo do apontador base.

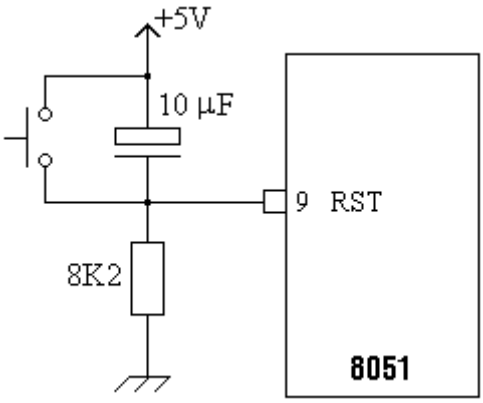
Assim o valor base do endereço do salto será carregado no apontador base (DPTR) e o valor de indexação do salto que realiza a condição (case) será carregado no acumulador.

Exemplo : JMP @ A + DPTR

Faz um salto para o endereço dador por A + DPTR

PC ← A + DPTR

7 - RESET NO 8051



O RESET é conseguido mantendo-se nível alto no pino 9 (RST) por pelo menos dois ciclos de máquina (24 pulsos de clock).

Para a versão CMOS (80C51) o resistor se torna desnecessário. Se existir não interfere em nada.

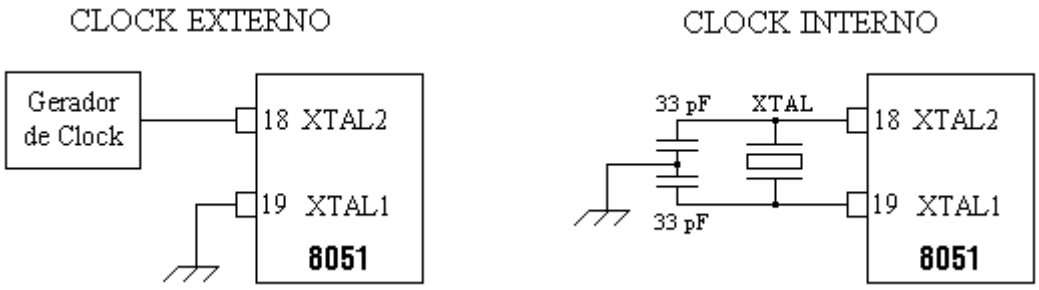
O RESET afetará os registros de função especial da seguinte forma:

REGISTRO	VALOR		REGISTRO	VALOR
PC	0000 H		TMOD	00 H
ACC	00 H		TCON	00 H
B	00 H		TH0	00 H
PSW	00H		TL0	00 H
SP	07 H		TH1	00 H
DPTR	0000 H		TL1	00 H
P0 a P3	FF H ou		SCON	00 H
	11111111 B		SBUF	indeterminado
IP	0XX00000 B		PCON (NMOS)	0XXXXXXX B
IE	00 H		PCON (CMOS)	0XXX0000 B

A RAM não é afetada por RESET forçado. Em caso de RESET por inicialização o seu valor será aleatório.

8 - CLOCK

Freqüência de clock mínima = 3,5 MHz
Freqüência de clock máxima = 33 MHz (algumas versões) e 12 MHz (mais comum)



9 - INTERRUPÇÕES NO 8051

O 8051 pode ser interrompido de 5 maneiras:

- pela interrupção externa 0 (INT0\ - pino P3.2);

- pela interrupção externa 1 (INT1\ - pino P3.3);

- pelo contador/temporizador 0;

- pelo contador/temporizador 1;

- pelo canal de comunicação serial.

As interrupções do 8051 são *VETORADAS*, ou seja, tem um endereço de início da rotina de tratamento da interrupção fixo.

Os vetores das interrupções são os seguintes:

INTERRUPÇÃO	Endereço no 8051	Endereço no kit
INT 0	0003 h	4230 h
C/T 0	000B h	4240 h
INT 1	0013 h	4250 h
C/T 1	001B h	4260 h
SERIAL	0023 h	4270 h

Quando atende uma interrupção, o PC é salvo no Stack (pilha). O acumulador, PSW e demais registros não são salvos.

As interrupções no 8051 são *MASCARADAS* , ou seja, podem ser desabilitadas pelo software. O registro responsável pelas habilitações da interrupções é o IE (Interrupt Enable).

AFh	AEh	ADh	ACh	Abh	AAh	A9h	A8h	A8 H
EA	x	x	ES	ET1	EX1	ET0	EX0	IE
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	

- EA (Enable All) - em "0" desabilita todas as interrupções;

em "1" permite que cada interupção seja habilitada individualmente.

- ES (Enable Serial) - Habilita ou desabilita a interrupção pedida pelo canal de serial;

"0" desabilita e "1" habilita se EA = 1;

- ET1 (Enable Timer 1) - Habilita ou desabilita a interrupção pedida pelo temporizador 1;

"0" desabilita e "1" habilita se EA = 1;

- EX1 (Enable External 1) - Habilita ou desabilita a interrupção externa 1;

"0" desabilita e "1" habilita se EA = 1;

- ET0 (Enable Timer 0) - Habilita ou desabilita a interrupção pedida pelo temporizador 0;

"0" desabilita e "1" habilita se EA = 1;

- EX0 (Enable External 0) - Habilita ou desabilita a interrupção externa 0;

"0" desabilita e "1" habilita se EA = 1;

Cada interrupção no 8051 poderá ser individualmente programada para um dos dois níveis de prioridade (0 ou 1). Isto é feito através dos bits do registro IP (Interrupt Priority).

BFh	BEh	BDh	BCh	BBh	BAh	B9h	B8h	B8 H
x	x	x	PS	PT1	PX1	PT0	PXO	IP
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	

"0" Prioridade BAIXA / "1" Prioridade ALTA

- PS (Priority Serial) - Nível de prioridade para o canal serial;

- PT1 (Priority Timer 1) - Nível de prioridade para o temporizador 1;

- PX1 (Priority External 1) - Nível de prioridade para a interrupção externa 1;

- PT0 (Priority Timer 0) - Nível de prioridade para o temporizador 0;

- PX0 (Priority External 0) - Nível de prioridade para a interrupção externa 0;

Uma interrupção de nível baixo poderá ser interrompida por outra de nível alto, mas não por outra de nível baixo. Uma interrupção de nível alto não poderá ser interrompida por qualquer outra fonte de interrupção.

Se ocorrerem dois pedidos de interrupções simultaneamente (quase impossível), e ambas forem de mesma prioridade, ocorrerá uma outra seleção interna, que escolherá qual interrupção será atendida primeiro.

INTERRUPÇÃO	PRIORIDADE INTERNA
Externa 0	MAIOR
Temporizador 0	
Externa 1	
Temporizador 1	
Serial	MENOR

As interrupções EXTERNAS no 8051 podem ser ativadas por *transição 1 para 0* (borda de descida) ou por *nível* (nível lógico baixo).

O ajuste da forma de ativar as interrupções externas estão no registro TCON (Controle do Temporizador).

8Fh	8Eh	8Dh	8Ch	8Bh	8Ah	89h	88h	88 H
TF1	TR1	TF0	TRO	IE1	IT1	IE0	IT0	TCON
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	

IT 0 = "0" - ativa a interrupção com nível lógico baixo no pino INT0\ (pino P3.2);

= "1" - ativa a interrupção com borda de descida no pino INT0\;

- IT 1 = "0" - ativa a interrupção com nível lógico baixo no pino INT1\ (pino P3.3);

= "1" - ativa a interrupção com borda de descida no pino INT1\;

- IE x - fica em "1"quando for detectado uma borda de descida (pedido de interrupção).

É resetado após o atendimento da rotina de tratamento da interrupção.

OBS.: Caso a interrupção seja ativada por nível lógico baixo, o pino INTx\ poderá permanecer em "0" durante a execução da rotina de tratamento da interrupção, só que deve estar em "1" antes do término da RTI para evitar um novo pedido de interrupção.

10 - TEMPORIZADORES E CONTADORES no 8051

O 8051 possui internamente 2 Contadores/Temporizadores denominados como TEMPORIZADOR 0 E TEMPORIZADOR 1. Ambos podem ser configurados para operar como temporizador ou contador de eventos, individualmente. Podem ter a operação habilitada por software ou hardware.

Na função de *temporizador*, um registro será incrementado a cada ciclo de máquina. Considerando que cada ciclo de máquina consiste em 12 períodos do clock, a taxa de contagem será de 1/12 da freqüência do clock.

Na função de *contador*, um registro será incrementado em resposta a uma transição de "1" para "0" (borda de descida) de seu correspondente pino de entrada externa, T0 (P3.4) e T1 (P3.5).

Nesta função, os pinos externos (T0 e T1) são amostrados a cada ciclo de máquina. Quando uma amostragem indicar um nível alto em um ciclo de máquina e um nível baixo no próximo ciclo, o contador será incrementado.

A máxima taxa de contagem será de 1/24 da freqüência do clock, visto que são necessárias dois ciclo de máquina para o reconhecimento de uma transição de "1" para "0".

A operação dos Contadores/Temporizadores terão quatro modos possíveis.

A seleção de Temporizador ou Contador, é realizada através do registro de função especial TMOD (Modo do Temporizador).

O registro TMOD é dividido em duas partes iguais que controlam o TEMPORIZADOR 1 e TEMPORIZADOR 0. O registro TMOD é acessado pelo endereço 89h e não é bit endereçável.

TMOD é o registro de controle de modo Temporizador / Contador, é neste registro que é feita a seleção de função Temporizador ou Contador e a seleção do modo de operação (modo 0, 1, 2 ou 3).

O registro TMOD é mostrado abaixo, assim como a função de cada bit.

GATE.1	C/T.1	M1.1	M0.1	GATE.0	C/T.O	M1.0	M0.0	89H
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
Controle do C/T 1				Controle do C/T 0				

C/T.x → Seleciona a função, TEMPORIZADOR (timer) ou CONTADOR. Será selecionado como TEMPORIZADOR se este bit estiver em "0". Se em "1" a operação será como CONTADOR.

GATE.x → Quando GATE.x = 1 e TRx = 1, o temporizador irá operar somente enquanto o pino INTx = 1 (controle por circuito). Quando GATE.x = 0, o temporizador irá operar somente quando TRx = 1 (controle por software).

Obs.: TRx é um bit de TCON (palavra de controle do Contador/Temporizador) que será vista a seguir.

M1.x e M0.x → Bits de seleção de modo de operação.

O registro **TCON** (Timer Control – Controle do Temporizador) é mostrado abaixo.

TCON é um registro acessado pelo endereço 88H e é bit endereçável.

8Fh	8Eh	8Dh	8Ch	8Bh	8Ah	89h	88h	TCON
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88H
TCON.7	TCON.6	TCON.5	TCON.4	TCON.3	TCON.2	TCON.1	TCON.0	

- **TFx** → Bit de overflow do temporizador. É ativado (setado) pelo circuito quando ocorrer um overflow no temporizador, gerando um pedido de interrupção. É ressetado pelo hardware após o terminada a rotina de interrupção.

- **TRx** → Bit de controle de operação do temporizador. É o bit que liga e desliga o C/T. Para ligar o temporizador, o software deverá setar este bit, e para desligar deverá ressetá-lo.

10.1 - Modos de Operação dos Contadores / Temporizadores

M 1.x	M 0.x	Modo de operação	Característica
0	0	Modo 0	C/T de 13 bits
0	1	Modo 1	C/T de 16 bits
1	0	Modo 2	C/T de 8 bits com recarga automática
1	1	Modo 3	1 C/T de 8 bits e 1 Temporizador

10.1.1 – MODO 0 – Contador / Temporizador de 8 bits com divisor de freqüência de até 32 vezes

Neste modo o registro do temporizador será configurado como um registro de 13 bits. Quando o registro (TH1 ou TH0) mudar de FFh para 00h ele habilita a solicitação de interrupção do temporizador, através de o bit TF1 ou TF0, do registro de função especial TCON (88h).

O registro de 13 bits é composto por todos os bits de TH1 ou TH0, e pelos 5 bits menos significativos de TL1 ou TL0. Estes 5 bits possibilitam a divisão por 32. Os 3 bits mais significativos de TL1 e TL0 são indeterminados e devem ser ignorados em caso de leitura. A habilitação de operação do temporizador através de TRx não limpa estes registros.

Os registros TH1, TH0, TL1 e TL0 estão localizados na região dos registros de função especial e são acessados pelos endereços 8Dh (TH1), 8Ch (TH0), 8Bh (TL1) e 8Ah (TL0). Estes registros não são bit endereçáveis.

No modo 0 a operação do temporizador poderá ser controlador por programação (software) ou por circuito externo (hardware).

O controle por circuito externo será realizado colocando-se o bit TRx em "1", o bit GATE.x em "1" e a entrada / INTx em "1". Desta maneira, será possível que a temporização seja controlada externamente por /INTx. Isto possibilita a medição de largura de pulsos, pois o temporizador somente opera enquanto o pino /INTx estiver em nível alto.

O controle por programação será realizado colocando-se o bit GATE.x em "0"e colocando-se TRx em "1". Neste caso o bit TRx atua como habilitador de operação do temporizador, que será controlado por programação.

Para fazermos um temporização, o registro THx receberá o valor inicial da contagem, e dessa forma podemos contar até FF, com o valor inicial escrito por software. Ao ocorrer overflow nesse registro (passar de FF para 00), o C/T em questão fará um pedido de interrupção, que será ou não aceito pela CPU. O valor deste registro poderá ser lido a qualquer momento pelo software.

O sinal de contagem (clock interno ou externo) será dividido pelo valor binário presente nos 5 bits menos significativos de TLx.

Desta maneira, para que o registro THx seja incrementado, deverá ter ocorrido um número de pulsos igual ao valor da contagem ajustada em TLx.

Quando ocorrer overflow (na troca de THx de FFh para 00h) um pedido de interrupção será feito. Caso a interrupção esteja habilitada, e deseja-se fazer uma nova temporização ou contagem, os registros THx e TLx deverão ser novamente carregados.

Como podemos dividir a freqüência no máximo por 32, e contar no máximo até 256, este modo permite-nos contar 8192 (256x32).

10.1.2 – MODO 1 – Contador / Temporizador de 16 bits

Neste modo de operação, o Contador/Temporizador será de 16 bits, sendo utilizado os registros TH1 e TL1 ou TH0 e TL0 para formar estes 16 bits. Desta forma podemos contar de 0000h até FFFFh (65536 contagens).

O valor inicial da contagem deverá ser programado por software. Quando ocorrer overflow (passar de FFFFh para 0000h) será feito um pedido de interrupção. Assim como no modo 0, o valor inicial de uma nova contagem deverá ser carregado na rotina de interrupção.

10.1.3 – MODO 2 – Contador / Temporizador de 8 bits com recarga automática

Neste modo, os registros TL1 e TL0 são responsáveis pela contagem e TH1 e TH0 ficam com o valor a ser carregado nos registros TL1 e TL0, quando ocorrer overflow. Mesmo com a recarga automática do valor da contagem, os registros TH1 e TH0 podem ser alterados a qualquer momento.

Neste modo de funcionamento, temos um sistema no qual não precisamos reescrever por software o valor a ser contado. Aqui, escrevemos no registro TLx, o valor inicial da contagem, e em THx o valor inicial da próxima contagem. Ao ocorrer o overflow em TLx, o sistema gera a interrupção e automaticamente recarrega em TLx o valor contido em THx, e prossegue incrementando sob comando do sinal externo (contador) ou interno (temporizador).

Devido a recarga automática, não é necessário que seja feita a carga do valor inicial na rotina de interrupção.

10.1.4 – MODO 3 – 1 Contador ou Temporizador de 8bits e 1 Temporizador de 8 bits

O modo 3 terá comportamentos diferentes para os dois temporizadores.

O temporizador 0 estabelece que TL0 e TH0 serão dois contadores separados.

TL0 usa os bits de controle do temporizador 0, ou seja, usa os bits C/T.0, GATE.0, TR0, TF0 e INT0 e pode operar como contador ou temporizador.

TH0 opera somente como temporizador e usa os bits de controle TR1 e TF1 do temporizador 1. Assim TH0 controla a interrupção do temporizador 1.

O temporizador 1 permanece com a sua contagem (mantém TH1 e TL1) quando no modo 3. Fica desligado, ou seja, é como se o TR1 estivesse em "0". Quando configurado para outro modo funciona normalmente, mas fica permanentemente ligado, ou seja, como se TR1 = "1".

O modo 3 atua como se existisse um temporizador extra de 8 bits. O temporizador 1 poderá ser usado em aplicações que não necessite de interrupções, como por exemplo, geração de taxa de transmissão para a porta serial.

> OBS.: Para ativar as interrupções dos Contadores/Temporizadores, os bits EA, ET1 e ET0 do registro IE (Interrupt Enable) deverão ser ajustados. Para habilitar a interrupção do C/T 1, além do bit EA =1, ET1 = 1 e para o C/T 0, EA =1 e ET0 = 1.

11 - O ASSEMBLER DO 8051

INSTRUÇÕES DO MICROCONTROLADOR 8051

* Rn → Indica RegistroR0 a R7 genericamente, dependente de "n".

* Ri → Indica RegistroR0 ou R1, dependendo de "i".

* @ → Significa "endereçado pelo valor de ...."

* #Dado → Indica valor constante de 8 bits.

* #Dado 16 → Indica valor constante de 16 bits.

* Direto → Indica um endereço de memória de 8 bits (256 posições internas - RAM interna e Registros de Função Especial).

* rel → Indica que endereçamento é relativo.

* ? → Indica que o flag indicado é afetado pela instrução e depende do resultado.

11.1 - INSTRUÇÕES DE TRANSFERÊNCIA DE DADOS

MOV A,Rn – Move o Registro *n* para o Acumulador. (1 byte – 12 pulsos);

MOV A,**Direto** – Move o conteúdo da posição de memória para o Acumulador. (2 bytes – 12 pulsos);

MOV A,@Ri – Move o conteúdo da RAM interna endereçada po Ri para o Acumulador. (1 byte – 12 pulsos);

MOV A,#Dado – Move o Dado para o Acumulador. (2 bytes – 12 pulsos);

MOV Rn,A – Move o conteúdo do Acumulador para o Registro *n*. (1 byte – 12 pulsos);

MOV Rn,**Direto** – Move o conteúdo da memória para o Registro *n*. (2 bytes – 24 pulsos);

MOV Rn,#Dado – Move o Dado para o Registro *n*. (2 bytes – 12 pulsos);

MOV* Direto*,A – Move o conteúdo do Acumulador para a posição de memória. (2 bytes – 12 pulsos);

MOV* Direto*,Rn – Move o conteúdo do Registro *n* para a posição de memória. (2 bytes – 24 pulsos);

MOV* Direto*1,*Direto*2 – Move o conteúdo da posição de memória 2 para a posição de memória 1. (3 bytes– 24 pulsos);

MOV* Direto*,@Ri – Move o conteúdo da posição de memória endereçada por Ri para a posição de memória. (2 bytes – 24 pulsos);

MOV* Direto*,#Dado – Move o Dado para a posição de memória. (3 bytes– 24 pulsos);

MOV @Ri,A – Move o conteúdo do Acumulador para a posição de memória endereçada por Ri. (1 byte – 12 pulsos);

MOV @Ri,**Direto** – Move o conteúdo da posição de memória para a posição de memória endereçada por Ri. (2 bytes – 24 pulsos);

MOV @Ri,#Dado – Move o Dado para a posição de memória endereçada por Ri. (2 bytes – 12 pulsos);

MOV DPTR,#Dado 16 – Move o Dado de 16 bits para o Registro DPTR. (3 bytes– 24 pulsos);

MOVC A,@A+DPTR – Move o conteúdo da posição de memória da ROM endereçada por A + DPTR. O endereço será de 16 bits. (1 byte – 24 pulsos);

MOVC A,@A+PC – Move o conteúdo da posição de memória da ROM endereçada por A + PC. O endereço será de 16 bits. (1 byte – 24 pulsos);

MOVX A,@Ri – Move o conteúdo da posição de memória da RAM externa endereçada por Ri para o Acumulador. (1 byte – 24 pulsos);

MOVX A,@DPTR – Move o conteúdo da posição de memória da RAM externa endereçada por DPTR para o Acumulador. (1 byte – 24 pulsos);

MOVX @Ri,A – Move o conteúdo do Acumulador para a posição de memória da RAM externa endereçada por Ri. (1 byte – 24 pulsos);

MOVX @DPTR,A – Move o conteúdo do Acumulador para a posição de memória da RAM externa endereçada por DPTR. (1 byte – 24 pulsos);

***PUSH* Direto** – Coloca na pilha o conteúdo da posição de memória. Incrementa o SP (Stack Pointer) e escreve na pilha. (2 bytes – 24 pulsos);

***POP* Direto** – Retira da pilha o Dado e coloca na posição de memória. (2 bytes – 24 pulsos);

XCH A,Rn – Troca entre si os conteúdo do Acumulador e do Registro *n*. (1 byte – 12 pulsos);

***XCH A,*Direto** – Troca entre si os conteúdo do Acumulador e do Registro *n*. (2 bytes – 12 pulsos);

XCH A,@Ri – Troca entre si os conteúdo do Acumulador e da posição de memória endereçada por Ri. (1 byte – 12 pulsos);

XCHD A,@Ri – Troca os nibbles menos significativos do conteúdo do Acumulador e da posição de memória endereçada por Ri. (1 byte – 12 pulsos).

11.2 - Instruções para Variáveis Booleanas

CLR C – Zera o Carry. (1 byte – 12 pulsos) (CY=0);

CLR Bit – Zera o bit Endereçado. (2 bytes – 12 pulsos);

SETB C – Seta o Carry. (1 byte – 12 pulsos) (CY=1);

SETB Bit – Seta o bit endereçado. (2 bytes – 12 pulsos);

CPL C – Complementa o Carry. (1 byte – 12 pulsos) (CY=?);

CPL Bit – Complementa o bit endereçado. (2 bytes – 12 pulsos);

ANL C,Bit – Operação AND entre o Carry e o bit endereçado. (2 bytes – 24 pulsos) (CY=?);

ANL C,/Bit – Operação AND entre o Carry e o complemento do bit endereçado. (2 bytes – 24 pulsos) (CY=?);

ORL C,Bit – Operação OR entre o Carry e o bit endereçado. (2 bytes – 24 pulsos) (CY=?);

ORL C,/Bit – Operação OR entre o Carry e o complemento do bit endereçado. (2 bytes – 24 pulsos) (CY=?);

MOV C,Bit – Move o bit endereçado para o Carry. (2 bytes – 12 pulsos) (CY=?)

MOV Bit,C – Move o Carry para o bit endereçado. (2 bytes – 24 pulsos);

JC rel – Salta se o Carry for "1". O jump é relativo. (2 bytes – 24 pulsos);

JNC rel – Salta se o Carry for "0". O jump é relativo. (2 bytes – 24 pulsos);

JB Bit, rel – Salta se o bit endereçado estiver em "1". (3 bytes – 24 pulsos);

JNB Bit,rel – Salta se o bit endereçado estiver em "0". (3 bytes – 24 pulsos);

JBC Bit,rel – Salta se o bit endereçado estiver em "1" depois zera o bit. (3 bytes– 24 pulsos).

11.3 - Instruções Aritméticas

ADD A,Rn – Soma o conteúdo do Registro *n* ao Acumulador. (1 byte – 12 pulsos) (CY=?, AC=?, OV=?);

ADD A,Direto – Soma o conteúdo da posição de memória ao Acumulador. (2 bytes – 12 pulsos) (CY=?, AC=?, OV=?);

ADD A,@Ri – Soma o conteúdo da posição de memória endereçada por Ri ao Acumulador. (1 byte – 12 pulsos) (CY=?, AC=?, OV=?);

ADD A,#Dado – Soma o Dado ao Acumulador. (2 bytes – 12 pulsos) (CY=?, AC=?, OV=?);

ADDC A,Rn – Soma o conteúdo do Registro *n* e o Carry ao Acumulador. (1 byte – 12 pulsos) (CY=?, AC=?, OV=?);

ADDC A,Direto – Soma o conteúdo da posição de memória e o Carry ao Acumulador. (2 bytes – 12 pulsos) (CY=?, AC=?, OV=?);

ADDC A,@Ri – Soma o conteúdo da posição de memória endereçada por Ri e o Carry ao Acumulador. (1 byte – 12 pulsos) (CY=?, AC=?, OV=?);

ADDC A,#Dado – Soma o Dado e o Carry ao Acumulador. (2 bytes – 12 pulsos) (CY=?, AC=?, OV=?);

SUBB A,Rn – Subtrai o conteúdo do Registro *n* e o Carry do Acumulador. (1 byte – 12 pulsos) (CY=?, AC=?, OV=?);

SUBB A,Direto – Subtrai o conteúdo da posição de memória e o Carry do Acumulador. (2 bytes – 12 pulsos) (CY=?, AC=?, OV=?);

SUBB A,@Ri – Subtrai o conteúdo da posição de memória endereçada por Ri e o Carry do Acumulador. (1 bytes – 12 pulsos) (CY=?, AC=?, OV=?);

SUBB A,#Dado – Subtrai o Dado e o Carry do Acumulador. (1 byte – 12 pulsos) (CY=?, AC=?, OV=?);

INC A – Incrementa o Acumulador. (1 byte – 12 pulsos);

INC Rn – Incrementa o Registro *n*. (1 byte – 12 pulsos);

INC Direto – Incrementa o conteúdo da posição de memória. (2 bytes – 12 pulsos);

INC @Ri – Incrementa o conteúdo da posição de memória endereçada por Ri. (1 byte – 12 pulsos);

DEC A – Decrementa o Acumulador. (1 byte – 12 pulsos).

DEC Rn – Decrementa o Registro *n*. (1 byte – 12 pulsos);

DEC Direto – Decrementa o conteúdo da posição de memória. (2 bytes – 12 pulsos);

DEC @Ri – Decrementa o conteúdo da posição de memória endereçada por Ri. (1 byte – 12 pulsos);

INC DPTR – Incrementa o DPTR. (1 byte – 24 pulsos);

MUL AB – Multiplica A e B. O resultado fica: parte mais significativa em B e menos significativa em Acumulador. (1 byte – 48 pulsos) (CY=0, OV=?);

DIV AB – Divide A e B. O resultado fica: a parte inteira no Acumulador e o resto em B. (1 byte – 48 pulsos) (CY=0, OV=0*) * OV=1 caso B inicialmente seja 00H;

DA A – Faz o ajuste decimal do acumulador. (1 byte – 12 pulsos) (CY=?, AC=?).

11.4 - INSTRUÇÕES DE DESVIO

ACALL End 11 – Chama sub-rotina numa faixa de 2 Kbytes da atual posição. (2 bytes – 24 pulsos);

LCALL End 16 – Chama sub-rotina em qualquer posição da memória de programa (ROM). (3 bytes – 24 pulsos);

RET – Retorno de sub-rotina. (1 byte – 24 pulsos);

RETI – Retorno de rotina de interrupção. (1 byte – 24 pulsos);

AJMP End 11 – Salta para outro endereço numa faixa de 2Kbytes da atual. (2 bytes – 24 pulsos);

LJMP End 16 – Salta para qualquer posição de memória de programa (ROM). (3 bytes – 24 pulsos);

SJMP rel – Salto curto relativo. Salta 127 posições para frente ou 128 para trás. (2 bytes – 24 pulsos);

JMP @A,DPTR – Salta para o endereço A + DPTR. (1 byte – 24 pulsos);

JZ rel – Salta se o Acumulador for zero. (2 bytes – 24 pulsos);

JNZ rel – Salta se o Acumulador não for zero. (2 bytes – 24 pulsos);

CJNE A, **Direto*****, rel*** – Compara e salta se o Acumulador for diferente da memória endereçada. (3 bytes – 24 pulsos) (CY=?);

CJNE A,#Dado,rel – Compara e salta se o Acumulador for diferente do Dado. (3 bytes – 24 pulsos) (CY=?);

CJNE Rn,#Dado,rel – Compara e salta se o Registro *n* for diferente do Dado. (3 bytes – 24 pulsos) (CY=?);

CJNE @Ri,#Dado,rel – Compara e salta se o conteúdo da RAM externa endereçada for diferente do Dado . (3 bytes – 24 pulsos) (CY=?);

DJNZ Rn,rel – Decrementa o Registro *n* e salta se for diferente de zero. (2 bytes – 24 pulsos);

DJNZ **Direto*****,rel*** – Decrementa o conteúdo da posição de memória e salta se for diferente de zero. (3 bytes – 24 pulsos);

NOP – Nenhuma operação. (1 byte – 12 pulsos).

11.5 - Instruções Lógicas

ANL A,Rn – Executa a operação AND entre o Registro *n* e o Acumulador. (1 byte – 12 pulsos);

ANL A,**Direto** – Executa a operação And entre o conteúdo da posição de memória e o Acumulador. (2 bytes – 12 pulsos);

ANL A,@Ri – Executa a operação AND entre o conteúdo da posição de memória endereçada por Ri. (1 byte – 12 pulsos);

ANL A,#Dado – Executa a operação AND entre o Dado e o Acumulador. (2 bytes – 12 pulsos);

ANL **Direto*****,A*** – Executa a operação AND entre o conteúdo da posição endereçada e Acumulador. (2 bytes – 12 pulsos);

ANL **Direto*****,#Dado***– Executa a operação AND entre a posição de memória endereçada e Dado. (3 bytes – 24 pulsos);

ORL A,Rn – Executa a operação OR entre o Registro *n* e o Acumulador. (1 byte – 12 pulsos);

ORL A,**Direto** – Executa a operação OR entre o conteúdo da posição de memória e o Acumulador. (2 bytes – 12

pulsos);

ORL A,@ Ri – Executa a operação OR entre o conteúdo da posição de memória endereçada por Ri. (1 byte – 12 pulsos);

ORL A,#Dado – Executa a operação OR entre o Dado e o Acumulador. (2 bytes – 12 pulsos);

ORL* Direto*,A – Executa a operação OR entre o conteúdo da posição endereçada e Acumulador. (2 bytes – 12 pulsos);

ORL* Direto*,#Dado – Executa a operação OR entre a posição de memória endereçada e o Dado. (3 bytes – 24 pulsos);

XRL A,Rn – Executa a operação "OU EXCLUSIVO" entre o Registro *n* e o Acumulador. (1 byte – 12 pulsos);

***XRL A,*Direto** – Executa a operação "OU EXCLUSIVO" entre o conteúdo da posição de memória e o Acumulador. O resultado fica no Acumulador. (2 bytes – 12 pulsos);

XRL A,@Ri – Executa a operação "OU EXCLUSIVO" entre o conteúdo da posição de memória endereçada por Ri e o Acumulador. (1 byte – 12 pulsos);

XRL A,#Dado – Executa a operação OU EXCLUSIVO" entre o Dado e o Acumulador. (2 bytes – 12 pulsos);

XRL Direto,A – Executa a operação "OU EXCLUSIVO" entre o conteúdo da posição de memória e o Acumulador O resultado fica na posição de memória. (2 bytes – 12 pulsos);

XRL Direto,#Dado – Executa a operação "OU EXCLUSIVO" entre o Dado e o conteúdo da posição de memória. O resultado fica na posição de memória. (3 bytes – 24 pulsos);

CLR A – Zera o Acumulador. (1 byte – 12 pulsos);

CPL A – Complementa o Acumulador. (1 byte – 12 pulsos);

RL A – Desloca o Acumulador à esquerda. (1 byte – 12 pulsos);

RLC A – Desloca o Acumulador à esquerda através do Carry. (1 byte – 12 pulsos) (CY=?);

RR A – Desloca o Acumulador à direita. (1 byte – 12 pulsos);

RRC A – Desloca o Acumulador à direita através do Carry. (1 byte – 12 pulsos) (CY=?);

SWAP A – Troca o nibble inferior do Acumulador com o superior. Equivale a 4 vezes RR A ou RL A. (1 byte – 12 pulsos).